University of Pittsburgh

Algorithms and Data Structures 2
CS 1501

Spring 2023

Sherif Khattab

ksm73@pitt.edu

Pitt SCI

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

Announcements

- Upcoming Deadlines
  - Homework 9: this Friday @ 11:59 pm
  - Lab 8: Tuesday 3/28 @ 11:59 pm
  - Assignment 3: Friday 3/31 @ 11:59 pm
    - Support video and slides on Canvas

Previous lecture

- ADT Graph
  - definitions
  - representations
  - traversals
    - BFS

This Lecture

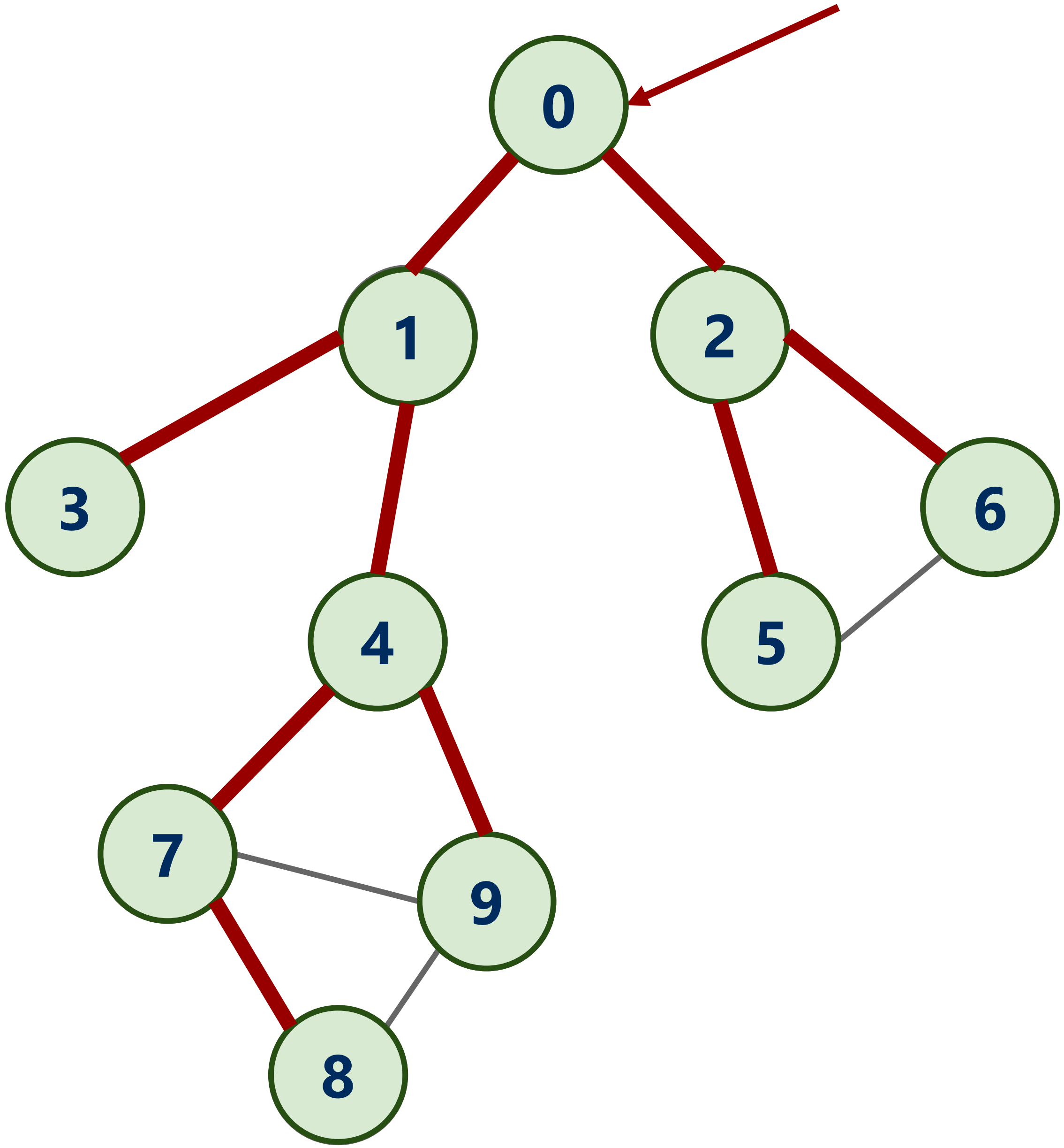
- ADT Graph
  - traversals
    - BFS
      - shortest paths based on number of edges
      - connected components
    - DFS
      - finding articulation points of a graph
  - Minimum Spanning Tree (MST) problem
    - Prim’s MST algorithm

BFS Pseudo-code

```
Q = new Queue

BFS(vertex v){
    add v to Q
    while(Q is not empty){
        w = remove head of Q
        visited[w] = true //mark w as visited
        for each unseen neighbor x
            seen[x] = true //mark x as seen
            parent[x] = w
            add x to Q
    }
}
```

BFS example



Shortest paths

- BFS traversals can further be used to determine the *shortest path* between two vertices

BFS Pseudo-code to compute shortest paths

```
Q = new Queue

BFS(vertex v){
    add v to Q
    while(Q is not empty){
        w = remove head of Q
        visited[w] = true //mark w as visited
        for each unseen neighbor x
            seen[x] = true //mark x as seen
            parent[x] = w
            distance[x] = distance[w] + 1
            add x to Q
    }
}
```

Problem of the Day

- **Input**: A file containing LinkedIn Connection information formatted like the following:
  - Account1: Connection1, Connection2, …
  - Account2: Connection1, Connection2, …
  - …

- **Output**: Answer the following questions:
  - Given two LI accounts, how "far" are they from each other?
    - E.g., 1st connection, 2nd connection, etc.
  - Are the accounts in the file all ***connected***?
    - If not, how many ***connected components*** are there?
  - Are there certain accounts that if removed, the remaining accounts become ***partitioned***?
    - These account are called ***articulation points***

Problem of the Day

- **Input**: A file containing LinkedIn Connection information formatted like the following:
  - Account1: Connection1, Connection2, …
  - Account2: Connection1, Connection2, …
  - …

- **Output**: Answer the following questions:
  - Given two LI accounts, how "far" are they from each other?
    - E.g., 1st connection, 2nd connection, etc.
  - Are the accounts in the file all ***connected***?
    - If not, how many ***connected components*** are there?
  - Are there certain accounts that if removed, the remaining accounts become ***partitioned***?
    - These account are called ***articulation points***

Finding connected components

- A connected component is a connected subgraph G′
  - (V′, E′)
    - $V' \subseteq V$
    - $E' = \{(u, v) \in E$ and both u and $v \in V'\}$
- To find all connected components:
  - wrapper function around BFS
  - A loop in the wrapper function will have to continually call bfs() while **there are still unseen vertices**
  - Each call will yield a spanning tree for a **connected component** of the graph

BFS Pseudo-code to compute connected components

```
int components = 0
for each vertex v in V
        if visited[v] = false
                components++
                Q = new Queue
                BFS(v)
```

```
BFS(vertex v){
        add v to Q
        component
        while(Q is not empty){
                w = remove head of Q
                visited[w] = true
                component[w] = components
                for each unseen neighbor x
                        seen[x] = true
                        add x to Q
        }
}
```

Problem of the Day

- **Input**: A file containing LinkedIn Connection information formatted like the following:
  - Account1: Connection1, Connection2, …
  - Account2: Connection1, Connection2, …
  - …

- **Output**: Answer the following questions:
  - Given two LI accounts, how "far" are they from each other?
    - E.g., 1st connection, 2nd connection, etc.
  - Are the accounts in the file all ***connected***?
    - If not, how many ***connected components*** are there?
  - Are there certain accounts that if removed, the remaining accounts become ***partitioned***?
    - These account are called ***articulation points***

Runtime Analysis of BFS

- Total time: **vertex processing time + edge processing tim**e
- Each vertex is added to the queue exactly once and removed exactly once
  - *v* add/remove operations
    - *O(v)* time for vertex processing
- Edges are processed when adding the list of neighbors to the queue

Runtime Analysis of BFS: Adjacency Lists

- Each edge is processed at most twice, one per edge endpoint
  - *O(e)* time for edge processing
- Total time: **vertex processing time + edge processing tim**e
  - *O(v + e)*

Runtime Analysis for BFS: Adjacency Matrix

- With Adjacency Matrix, BFS checks each *possible* edge!
    - $O(v^2)$ time for edge processing
- Total time: **vertex processing time + edge processing tim**e
    - $O(v^2 + v) = O(v^2)$
- ***Running time depends on data structure selection!***

Problem of the Day

- **Input**: A file containing LinkedIn Connection information formatted like the following:
  - Account1: Connection1, Connection2, …
  - Account2: Connection1, Connection2, …
  - …

- **Output**: Answer the following questions:
  - Given two LI accounts, how "far" are they from each other?
    - E.g., 1st connection, 2nd connection, etc.
  - Are the accounts in the file all ***connected***?
    - If not, how many ***connected components*** are there?
  - Are there certain accounts that if removed, the remaining accounts become ***partitioned***?
    - These account are called ***articulation points***

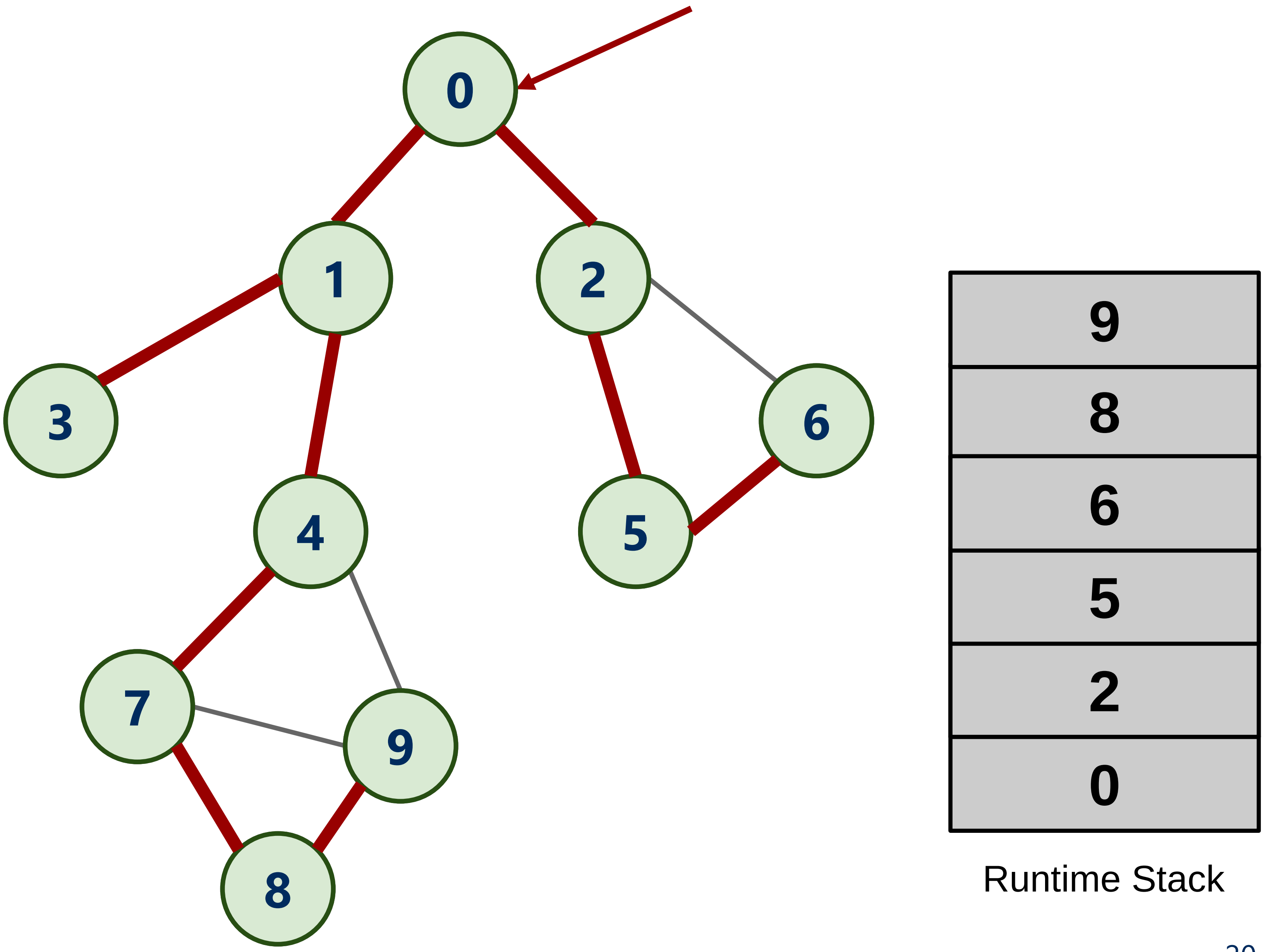
DFS – Depth First Search

- Already seen and used this throughout the term
  - For Huffman encoding…
    - as we build the codebook from the Huffman Trie
- Can be easily implemented recursively
  - For each vertex, visit *first* unseen neighbor
  - Backtrack at deadends (i.e., vertices with no unseen neighbors)
    - Try *next* unseen neighbor after backtracking
  - An arbitrary order of neighbors is assumed

DFS Pseudo-code

```
DFS(vertex v) {
  seen[v] = true //mark v as seen
  for each unseen neighbor w
    parent[w] = v
    DFS(w)
}
```

DFS example



When to visit a vertex

```
DFS(vertex v) {
   seen[v] = true //mark v as seen
   visit v //pre-order DFS
   for each unseen neighbor w
      parent[w] = v
      DFS(w)
}
```

When to visit a vertex

```
DFS(vertex v) {
   seen[v] = true //mark v as seen
for each unseen neighbor w
      parent[w] = v
      DFS(w)
 visit v //post-order DFS
}
```

When to visit a vertex

```
DFS(vertex v) {
   seen[v] = true //mark v as seen
for each unseen neighbor w
      parent[w] = v
      DFS(w)
      (re)visit v //in-order DFS
}
```

Runtime Analysis of DFS: Adjacency Lists

- Total time: **vertex processing time + edge processing time**
- Each vertex is seen then visited exactly once
  - *O(v)* time for vertex processing
  - except for in-order DFS
    - vertex processing is included in edge processing in that case
- Edges are processed when finding the list of neighbors
- Each edge is checked at most twice, one per edge endpoint
  - *O(e)* time for edge processing
- Total time: *O(v + e)*

Runtime Analysis of BFS and DFS

- At a high level, DFS and BFS have the same runtime
  - Each vertex must be seen and then visited, but the order will differ between these two approaches
- The representation of the graph affect the runtimes of of these traversal algorithms?
  - $O(v + e)$ with Adjacency Lists
  - $O(v^2)$ with Adjacency Matrix
  - Note that for a dense graph, $v + e = O(v^2)$

Problem of the Day

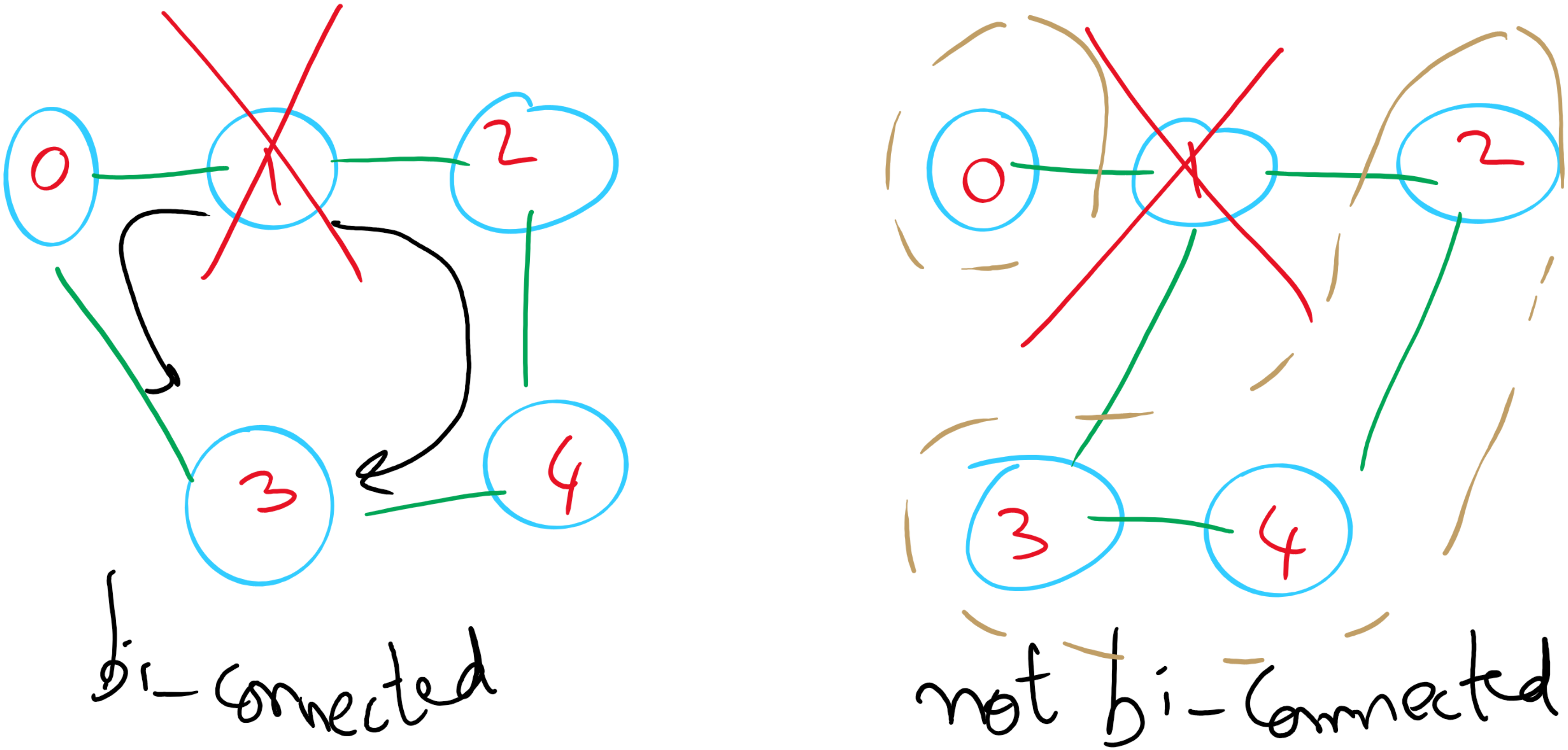
- **Input**: A file containing LinkedIn Connection information formatted like the following:
  - Account1: Connection1, Connection2, …
  - Account2: Connection1, Connection2, …
  - …

- **Output**: Answer the following questions:
  - Given two LI accounts, how "far" are they from each other?
    - E.g., 1st connection, 2nd connection, etc.
  - Are the accounts in the file all ***connected***?
    - If not, how many ***connected components*** are there?
  - Are there certain accounts that if removed, the remaining accounts become ***partitioned***?
    - These account are called ***articulation points***

Biconnected graphs

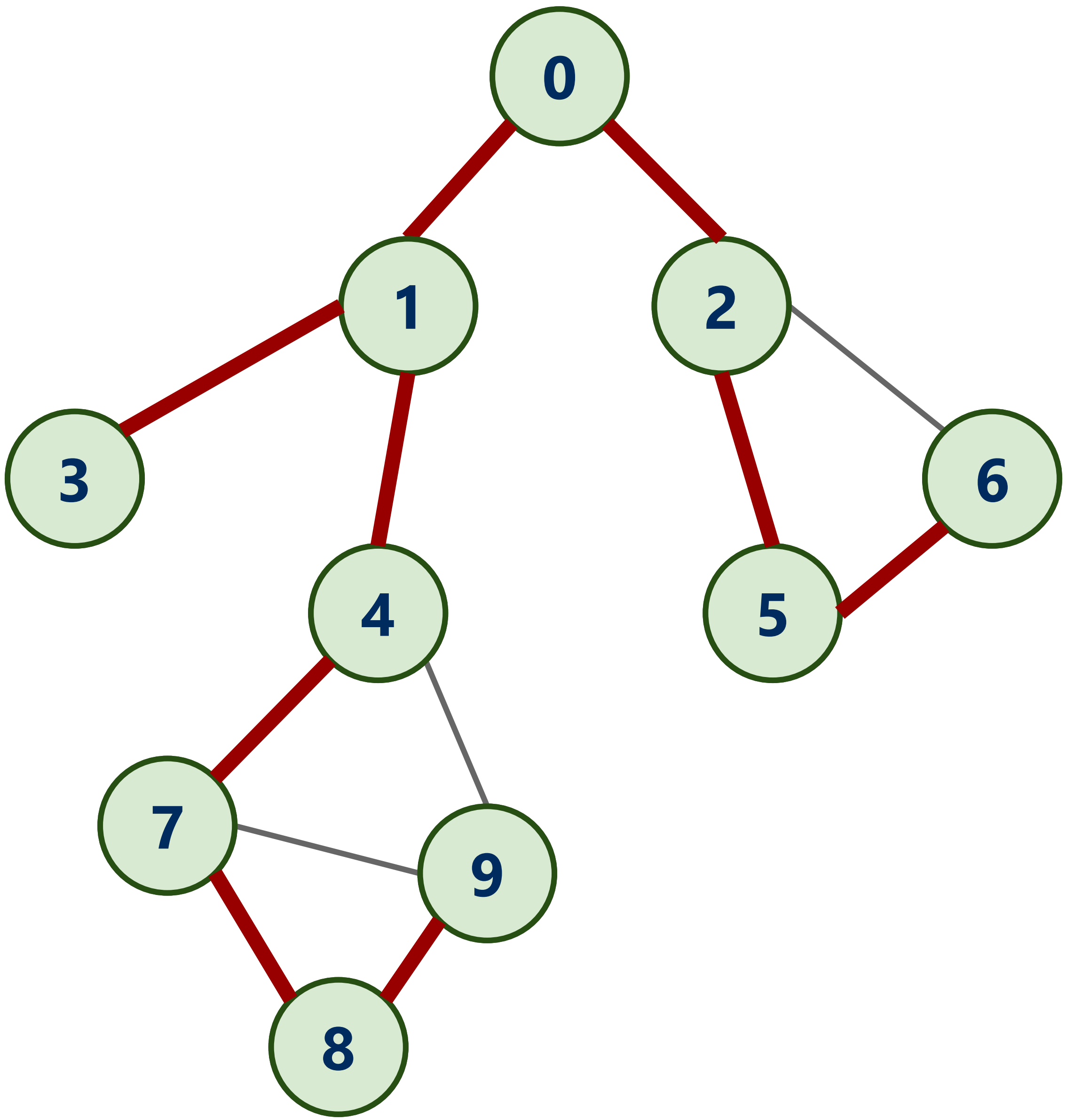
- A *biconnected graph* has at least 2 distinct paths between all vertex pairs
  - a distinct path shares no common edges or vertices with another path except for the start and end vertices
- A graph is biconnected graph iff it has zero *articulation points*
  - Vertices, that, if removed, will separate the graph

Biconnected Graph



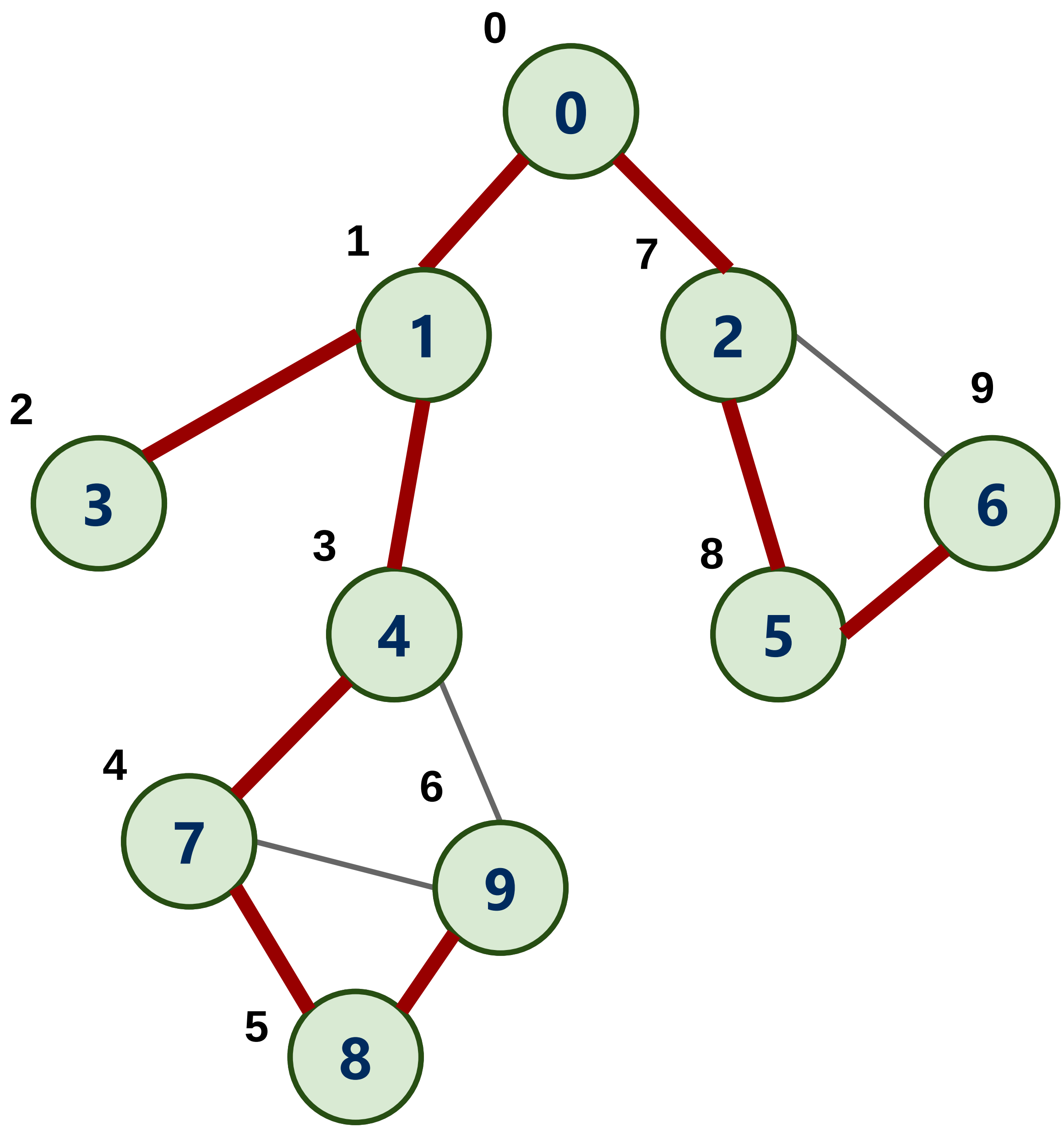
Finding articulation points of a graph

- A DFS traversal builds a spanning tree
  - red edges in the picture
- Edges not included in the spanning tree are called **back edges**
  - e.g., (4, 9) and (2, 6)



num(v)

- A pre-order DFS traversal visits the vertices in some order
  - let’s number the vertices with their traversal order
  - num(v)



Finding articulation points of a graph

- For each non-root vertex v, find the lowest numbered vertex reachable from v
  - **not through v's parent**
  - **using 0 or more tree edges then at most one back edge**
- move down the tree looking for a back edge that goes backwards the furtheset